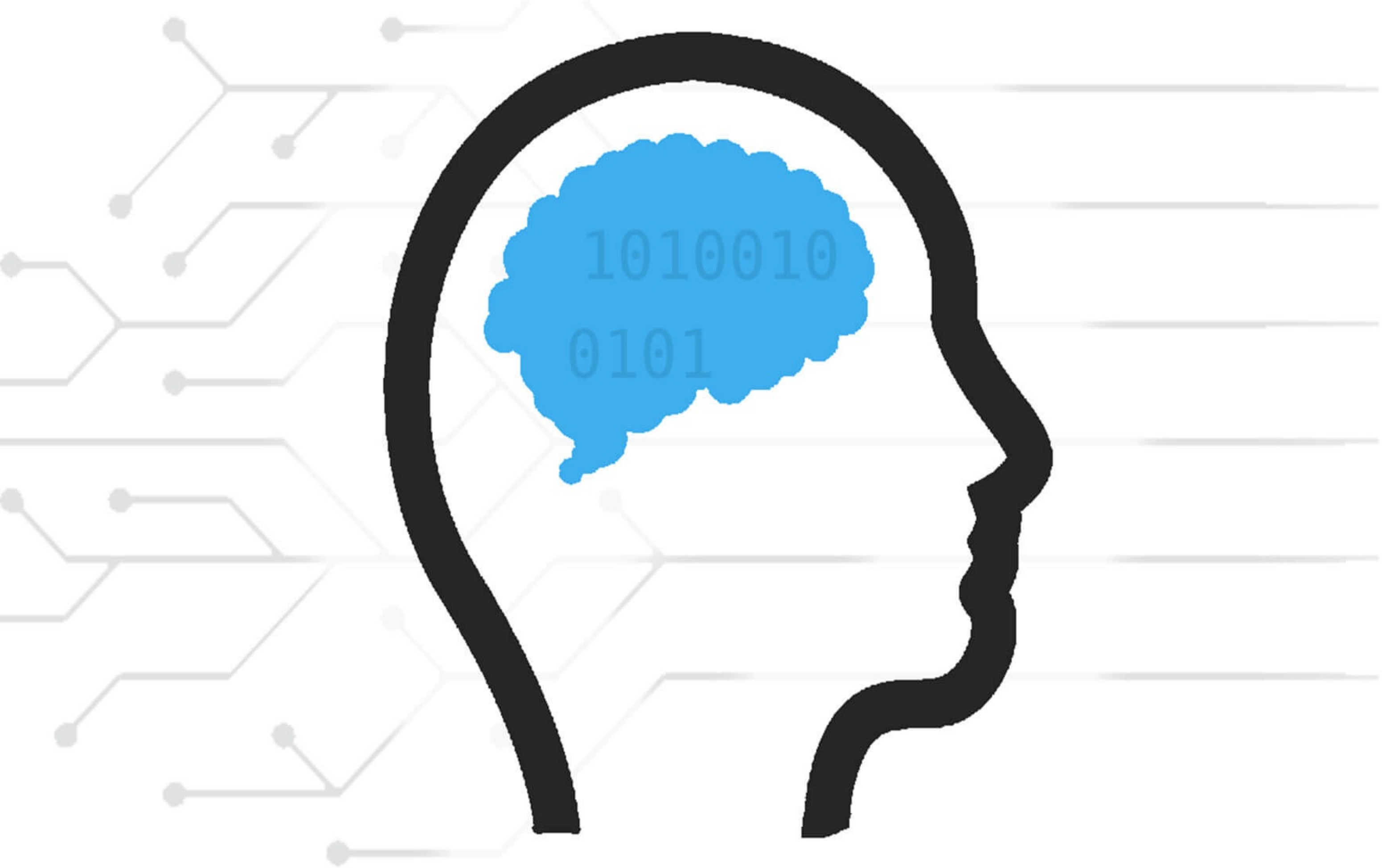

Algorithm

🄯 *http://algorithm.yuanbin.me*

Table of Contents

Preface	1.1
FAQ	1.2
Guidelines for Contributing	1.2.1
Contributors	1.2.2
Part I - Basics	1.3
Basics Data Structure	1.4
String	1.4.1
Linked List	1.4.2
Binary Tree	1.4.3
Huffman Compression	1.4.4
Queue	1.4.5
Heap	1.4.6
Stack	1.4.7
Set	1.4.8
Map	1.4.9
Graph	1.4.10
Basics Sorting	1.5
Bubble Sort	1.5.1
Selection Sort	1.5.2
Insertion Sort	1.5.3
Merge Sort	1.5.4
Quick Sort	1.5.5
Heap Sort	1.5.6
Bucket Sort	1.5.7
Counting Sort	1.5.8
Radix Sort	1.5.9
Basics Algorithm	1.6
Divide and Conquer	1.6.1
Binary Search	1.6.2
Math	1.6.3
Greatest Common Divisor	1.6.3.1
Prime	1.6.3.2
Knapsack	1.6.4
Counting Problem	1.6.5
Probability	1.6.6
Shuffle	1.6.6.1
Bitmap	1.6.7
Basics Misc	1.7
Bit Manipulation	1.7.1

Part II - Coding 1.8
String 1.9
- strStr 1.9.1
- Two Strings Are Anagrams 1.9.2
- Compare Strings 1.9.3
- Anagrams 1.9.4
- Longest Common Substring 1.9.5
- Rotate String 1.9.6
- Reverse Words in a String 1.9.7
- Valid Palindrome 1.9.8
- Longest Palindromic Substring 1.9.9
- Space Replacement 1.9.10
- Wildcard Matching 1.9.11
- Length of Last Word 1.9.12
- Count and Say 1.9.13

Integer Array 1.10
- Remove Element 1.10.1
- Zero Sum Subarray 1.10.2
- Subarray Sum K 1.10.3
- Subarray Sum Closest 1.10.4
- Recover Rotated Sorted Array 1.10.5
- Product of Array Exclude Itself 1.10.6
- Partition Array 1.10.7
- First Missing Positive 1.10.8
- 2 Sum 1.10.9
- 3 Sum 1.10.10
- 3 Sum Closest 1.10.11
- Remove Duplicates from Sorted Array 1.10.12
- Remove Duplicates from Sorted Array II 1.10.13
- Merge Sorted Array 1.10.14
- Merge Sorted Array II 1.10.15
- Median 1.10.16
- Partition Array by Odd and Even 1.10.17
- Kth Largest Element 1.10.18

Binary Search 1.11
- Binary Search 1.11.1
- Search Insert Position 1.11.2
- Search for a Range 1.11.3
- First Bad Version 1.11.4
- Search a 2D Matrix 1.11.5
- Search a 2D Matrix II 1.11.6
- Find Peak Element 1.11.7

- Search in Rotated Sorted Array 1.11.8
- Search in Rotated Sorted Array II 1.11.9
- Find Minimum in Rotated Sorted Array 1.11.10
- Find Minimum in Rotated Sorted Array II 1.11.11
- Median of two Sorted Arrays 1.11.12
- Sqrt x 1.11.13
- Wood Cut 1.11.14

Math and Bit Manipulation 1.12

- Single Number 1.12.1
- Single Number II 1.12.2
- Single Number III 1.12.3
- O1 Check Power of 2 1.12.4
- Convert Integer A to Integer B 1.12.5
- Factorial Trailing Zeroes 1.12.6
- Unique Binary Search Trees 1.12.7
- Update Bits 1.12.8
- Fast Power 1.12.9
- Hash Function 1.12.10
- Count 1 in Binary 1.12.11
- Fibonacci 1.12.12
- A plus B Problem 1.12.13
- Print Numbers by Recursion 1.12.14
- Majority Number 1.12.15
- Majority Number II 1.12.16
- Majority Number III 1.12.17
- Digit Counts 1.12.18
- Ugly Number 1.12.19
- Plus One 1.12.20

Linked List 1.13

- Remove Duplicates from Sorted List 1.13.1
- Remove Duplicates from Sorted List II 1.13.2
- Remove Duplicates from Unsorted List 1.13.3
- Partition List 1.13.4
- Add Two Numbers 1.13.5
- Two Lists Sum Advanced 1.13.6
- Remove Nth Node From End of List 1.13.7
- Linked List Cycle 1.13.8
- Linked List Cycle II 1.13.9
- Reverse Linked List 1.13.10
- Reverse Linked List II 1.13.11
- Merge Two Sorted Lists 1.13.12
- Merge k Sorted Lists 1.13.13

Reorder List 1.13.14
Copy List with Random Pointer 1.13.15
Sort List 1.13.16
Insertion Sort List 1.13.17
Palindrome Linked List 1.13.18
Delete Node in the Middle of Singly Linked List 1.13.19
LRU Cache 1.13.20
Rotate List 1.13.21
Swap Nodes in Pairs 1.13.22
Remove Linked List Elements 1.13.23
Binary Tree 1.14
Binary Tree Preorder Traversal 1.14.1
Binary Tree Inorder Traversal 1.14.2
Binary Tree Postorder Traversal 1.14.3
Binary Tree Level Order Traversal 1.14.4
Binary Tree Level Order Traversal II 1.14.5
Maximum Depth of Binary Tree 1.14.6
Balanced Binary Tree 1.14.7
Binary Tree Maximum Path Sum 1.14.8
Lowest Common Ancestor 1.14.9
Invert Binary Tree 1.14.10
Diameter of a Binary Tree 1.14.11
Construct Binary Tree from Preorder and Inorder Traversal 1.14.12
Construct Binary Tree from Inorder and Postorder Traversal 1.14.13
Subtree 1.14.14
Binary Tree Zigzag Level Order Traversal 1.14.15
Binary Tree Serialization 1.14.16
Binary Search Tree 1.15
Insert Node in a Binary Search Tree 1.15.1
Validate Binary Search Tree 1.15.2
Search Range in Binary Search Tree 1.15.3
Convert Sorted Array to Binary Search Tree 1.15.4
Convert Sorted List to Binary Search Tree 1.15.5
Binary Search Tree Iterator 1.15.6
Exhaustive Search 1.16
Subsets 1.16.1
Unique Subsets 1.16.2
Permutations 1.16.3
Unique Permutations 1.16.4
Next Permutation 1.16.5
Previous Permuation 1.16.6
Permutation Index 1.16.7

Permutation Index II 1.16.8
Permutation Sequence 1.16.9
Unique Binary Search Trees II 1.16.10
Palindrome Partitioning 1.16.11
Combinations 1.16.12
Combination Sum 1.16.13
Combination Sum II 1.16.14
Minimum Depth of Binary Tree 1.16.15
Word Search 1.16.16
Dynamic Programming 1.17
Triangle 1.17.1
Backpack 1.17.2
Backpack II 1.17.3
Minimum Path Sum 1.17.4
Unique Paths 1.17.5
Unique Paths II 1.17.6
Climbing Stairs 1.17.7
Jump Game 1.17.8
Word Break 1.17.9
Longest Increasing Subsequence 1.17.10
Palindrome Partitioning II 1.17.11
Longest Common Subsequence 1.17.12
Edit Distance 1.17.13
Jump Game II 1.17.14
Best Time to Buy and Sell Stock 1.17.15
Best Time to Buy and Sell Stock II 1.17.16
Best Time to Buy and Sell Stock III 1.17.17
Best Time to Buy and Sell Stock IV 1.17.18
Distinct Subsequences 1.17.19
Interleaving String 1.17.20
Maximum Subarray 1.17.21
Maximum Subarray II 1.17.22
Longest Increasing Continuous subsequence 1.17.23
Longest Increasing Continuous subsequence II 1.17.24
Egg Dropping Puzzle 1.17.25
Maximal Square 1.17.26
Graph 1.18
Find the Connected Component in the Undirected Graph 1.18.1
Route Between Two Nodes in Graph 1.18.2
Topological Sorting 1.18.3
Word Ladder 1.18.4
Bipartial Graph Part I 1.18.5

Data Structure 1.19
- Implement Queue by Two Stacks 1.19.1
- Min Stack 1.19.2
- Sliding Window Maximum 1.19.3
- Longest Words 1.19.4
- Heapify 1.19.5
- Kth Smallest Number in Sorted Matrix 1.19.6

Problem Misc 1.20
- Nuts and Bolts Problem 1.20.1
- String to Integer 1.20.2
- Insert Interval 1.20.3
- Merge Intervals 1.20.4
- Minimum Subarray 1.20.5
- Matrix Zigzag Traversal 1.20.6
- Valid Sudoku 1.20.7
- Add Binary 1.20.8
- Reverse Integer 1.20.9
- Gray Code 1.20.10
- Find the Missing Number 1.20.11
- N Queens 1.20.12
- N Queens II 1.20.13
- Minimum Window Substring 1.20.14
- Continuous Subarray Sum 1.20.15
- Continuous Subarray Sum II 1.20.16
- Longest Consecutive Sequence 1.20.17

Part III - Contest 1.21

Google APAC 1.22
- APAC 2015 Round B 1.22.1
  - Problem A. Password Attacker 1.22.1.1
- APAC 2016 Round D 1.22.2
  - Problem A. Dynamic Grid 1.22.2.1

Microsoft 1.23
- Microsoft 2015 April 1.23.1
  - Problem A. Magic Box 1.23.1.1
  - Problem B. Professor Q's Software 1.23.1.2
  - Problem C. Islands Travel 1.23.1.3
  - Problem D. Recruitment 1.23.1.4
- Microsoft 2015 April 2 1.23.2
  - Problem A. Lucky Substrings 1.23.2.1
  - Problem B. Numeric Keypad 1.23.2.2
  - Problem C. Spring Outing 1.23.2.3
- Microsoft 2015 September 2 1.23.3

Problem A. Farthest Point 1.23.3.1
Appendix I Interview and Resume 1.24
Interview 1.24.1
Resume 1.24.2
Appendix II System Design 1.25
The System Design Process 1.25.1
Statistics 1.25.2
System Architecture 1.25.3
Scalability 1.25.4
Tags 1.26

Data Structure and Algorithm/leetcode/lintcode

build passing slack 1/303 chat on slack

- English via Data Structure and Algorithm notes
- 简体中文请戳 数据结构与算法/leetcode/lintcode题解
- 繁體中文請瀏覽 資料結構與演算法/leetcode/lintcode題解

Introduction

This work is some notes of learning and practicing data structures and algorithm.

1. Part I is some brief introduction of basic data structures and algorithm, such as, linked lists, stack, queues, trees, sorting and etc.
2. Part II is the analysis and summary of programming problems, and most of the programming problems come from https://leetcode.com/, http://www.lintcode.com/, http://www.geeksforgeeks.org/, http://hihocoder.com/, https://www.topcoder.com/.
3. Part III is the appendix of resume and other supplements.

This project is hosted on https://github.com/billryan/algorithm-exercise and rendered by Gitbook. You can star the repository on the GitHub to keep track of updates. Another choice is to subscribe channel `#github_commit` via Slack https://ds-algo.slack.com/messages/github_commit/. ~~RSS feed is under development.~~

Feel free to access http://slackin4ds-algo.herokuapp.com for Slack invite automation.

You can view/search this document online or offline, feel free to read it. :)

- Online(Rendered by Gitbook): http://algorithm.yuanbin.me
- Offline(Compiled by Gitbook and Travis-CI):
  1. EPUB: GitHub, Gitbook, 七牛 CDN(中国大陆用户适用) - Recommended for iPhone/iPad/MAC
  2. PDF: GitHub, Gitbook, 七牛 CDN(中国大陆用户适用) - Recommended for Desktop
  3. MOBI: GitHub, Gitbook, 七牛 CDN(中国大陆用户适用) - Recommended for Kindle
- Site Search via Google: `keywords site:algorithm.yuanbin.me`
- Site Search via Swiftype: Click `Search this site` on the right bottom of webpages

License

This work is licensed under the **Creative Commons Attribution-ShareAlike 4.0 International License**. To view a copy of this license, please visit http://creativecommons.org/licenses/by-sa/4.0/

Contribution

- English is maintained by @billryan
- 简体中文 is maintained by @billryan, @Shaunwei
- 繁體中文 is maintained by @CrossLuna

Other contributors can be found in Contributors to algorithm-exercise

Donation

本项目接受捐赠，形式不限，可以买书，可以寄明信片，也可以金额打赏 :)

邮寄明信片

@billryan 喜欢收集各种明信片，来者不拒~ 邮寄的话可以邮寄至 `上海市闵行区上海交通大学闵行校区电院群楼5号楼307` ，收件人： `袁斌` 。

送书

除了邮寄明信片，你还可以买本书送给各位贡献者，@billryan 的地址见上节。

支付宝

账户名：yuanbin2014(at)gmail.com 金额随意

Wechat

金额随意

PayPal

账户名：yuanbin2014(at)gmail.com 金额随意，付款时选择 friends and family

隐私考虑，以下名单隐去了部分个人信息，有些名单若没来得及添加，可私下联系我加上，有些信息和金额因为时间久远可能有误，欢迎指正。

- `taoli***@gmail.com` , 20
- `张亚*` , 6.66
- `wen***@126.com` , 20.16

- `she***@163.com` , 10
- `孙*` , 20
- `石*` , 50
- `文*` , 20
- `don***@163.com` , 5
- `129***@qq.com` , 50
- `130****9675` , 5
- `Tong W***` , 20 $
- `ee.***@gmail.com` , 6.66

所得捐款用于七牛 CDN 流量付费/激励 Contributors 写出更好的内容/购买书籍/西瓜/饮料

To Do

- [ ] add multiple languages support, currently 繁體中文, 简体中文 are available
- [x] explore nice writing style
- [x] add implementations of `Python` , `C++` , `Java` code
- [x] add time and space complexity analysis
- [x] summary of basic data structure and algorithm
- [x] add CSS for online website http://algorithm.yuanbin.me
- [x] add proper Chinese fonts for PDF output

FAQ - Frequently Asked Question

Some guidelines for contributing and other questions are listed here.

How to Contribute?

- Access Guidelines for Contributing for details.

Guidelines for Contributing

- Access English via Guidelines for Contributing
- 繁體中文請移步 貢獻指南
- 简体中文请移步 贡献指南

Part I - Basics

The first part summarizes some of the main aspects of data structures and algorithms, such as implementation and usage.

This chapter consists of the following sections.

Reference

- VisuAlgo - Animated visualizations of data structures and algorithms
- Data Structure Visualizations - An alternative to VisuAlgo
- Sorting Algorithms - Animations comparing various sorting algorithms

Data Structure

This chapter describes the fundamental data structures and their implementations.

String

String-related problems often appear in interview questions. In actual development, strings are also frequently used. Summarized here are common uses of strings in C++, Java, and Python.

Python

```
s1 = str()
# in python, `''` and `""` are the same
s2 = "shaunwei" # 'shaunwei'
s2len = len(s2)
# last 3 chars
s2[-3:] # wei
s2[5:8] # wei
s3 = s2[:5] # shaun
s3 += 'wei' # return 'shaunwei'
# list in python is same as ArrayList in java
s2list = list(s3)
# string at index 4
s2[4] # 'n'
# find index at first
s2.index('w')  # return 5, if not found, throw ValueError
s2.find('w') # return 5, if not found, return -1
```

In Python, there's no StringBuffer or StringBuilder. However, string manipulations are fairly efficient already.

Java

```
String s1 = new String();
String s2 = "billryan";
int s2Len = s2.length();
s2.substring(4, 8); // return "ryan"
StringBuilder s3 = new StringBuilder(s2.substring(4, 8));
s3.append("bill");
String s2New = s3.toString(); // return "ryanbill"
// convert String to char array
char[] s2Char = s2.toCharArray();
// char at index 4
char ch = s2.charAt(4); // return 'r'
// find index at first
int index = s2.indexOf('r'); // return 4. if not found, return -1
```

The difference between StringBuffer and StringBuilder is that the former guarantees thread safety. In a single-threaded environment, StringBuilder is more efficient.

String

String related topics are discussed in this chapter.

> In order to re-use most of the memory of an existing data structure, internal implementation of string is immutable in most programming languages(Java, Python). Take care if you want to modify character in place.

strStr

Question

- leetcode: Implement strStr() | LeetCode OJ
- lintcode: lintcode - (13) strstr

Problem Statement

For a given source string and a target string, you should output the **first** index(from 0) of target string in source string.

If target does not exist in source, just return `-1`.

Example

If source = `"source"` and target = `"target"`, return `-1`.

If source = `"abcdabcdefg"` and target = `"bcd"`, return `1`.

Challenge

O(n2) is acceptable. Can you implement an O(n) algorithm? (hint: *KMP*)

Clarification

Do I need to implement KMP Algorithm in a real interview?

- Not necessary. When you meet this problem in a real interview, the interviewer may just want to test your basic implementation ability. But make sure your confirm with the interviewer first.

Problem Analysis

It's very straightforward to solve string match problem with nested for loops. Since we must iterate the target string, we can optimize the iteration of source string. It's unnecessary to iterate the source string if the length of remaining part does not exceed the length of target string. We can only iterate the valid part of source string. Apart from this naive algorithm, you can use a more effective algorithm such as KMP.

Python

```python
class Solution:
    def strStr(self, source, target):
        if source is None or target is None:
            return -1

        for i in range(len(source) - len(target) + 1):
            for j in range(len(target)):
                if source[i + j] != target[j]:
                    break
            else:  # no break
                return i
        return -1
```

C

```c
int strStr(char* haystack, char* needle) {
    if (haystack == NULL || needle == NULL) return -1;

    const int len_h = strlen(haystack);
    const int len_n = strlen(needle);
    for (int i = 0; i < len_h - len_n + 1; i++) {
        int j = 0;
        for (; j < len_n; j++) {
            if (haystack[i+j] != needle[j]) {
                break;
            }
        }
        if (j == len_n) return i;
    }

    return -1;
}
```

C++

```cpp
class Solution {
public:
    int strStr(string haystack, string needle) {
        if (haystack.empty() && needle.empty()) return 0;
        if (haystack.empty()) return -1;
        if (needle.empty()) return 0;
        // in case of overflow for negative
        if (haystack.size() < needle.size()) return -1;

        for (int i = 0; i < haystack.size() - needle.size() + 1; i++) {
            string::size_type j = 0;
            for (; j < needle.size(); j++) {
                if (haystack[i + j] != needle[j]) break;
            }
            if (j == needle.size()) return i;
        }

        return -1;
    }
};
```

Java

```java
public class Solution {
    public int strStr(String haystack, String needle) {
        if (haystack == null && needle == null) return 0;
        if (haystack == null) return -1;
        if (needle == null) return 0;

        for (int i = 0; i < haystack.length() - needle.length() + 1; i++) {
            int j = 0;
            for (; j < needle.length(); j++) {
                if (haystack.charAt(i+j) != needle.charAt(j)) break;
            }
            if (j == needle.length()) return i;
        }

        return -1;
    }
}
```

Source Code Analysis

1. corner case: `haystack(source)` and `needle(target)` may be empty string.

2. code convention:
    - space is needed for `==`
    - use meaningful variable names
    - put a blank line before declaration `int i, j;`
3. declare j outside for loop if and only if you want to use it outside.

Some Pythonic notes: 4. More Control Flow Tools section 4.4 and if statement - Why does python use 'else' after for and while loops?

Complexity Analysis

nested for loop, $O((n-m)m)$ for worst case.

Tags